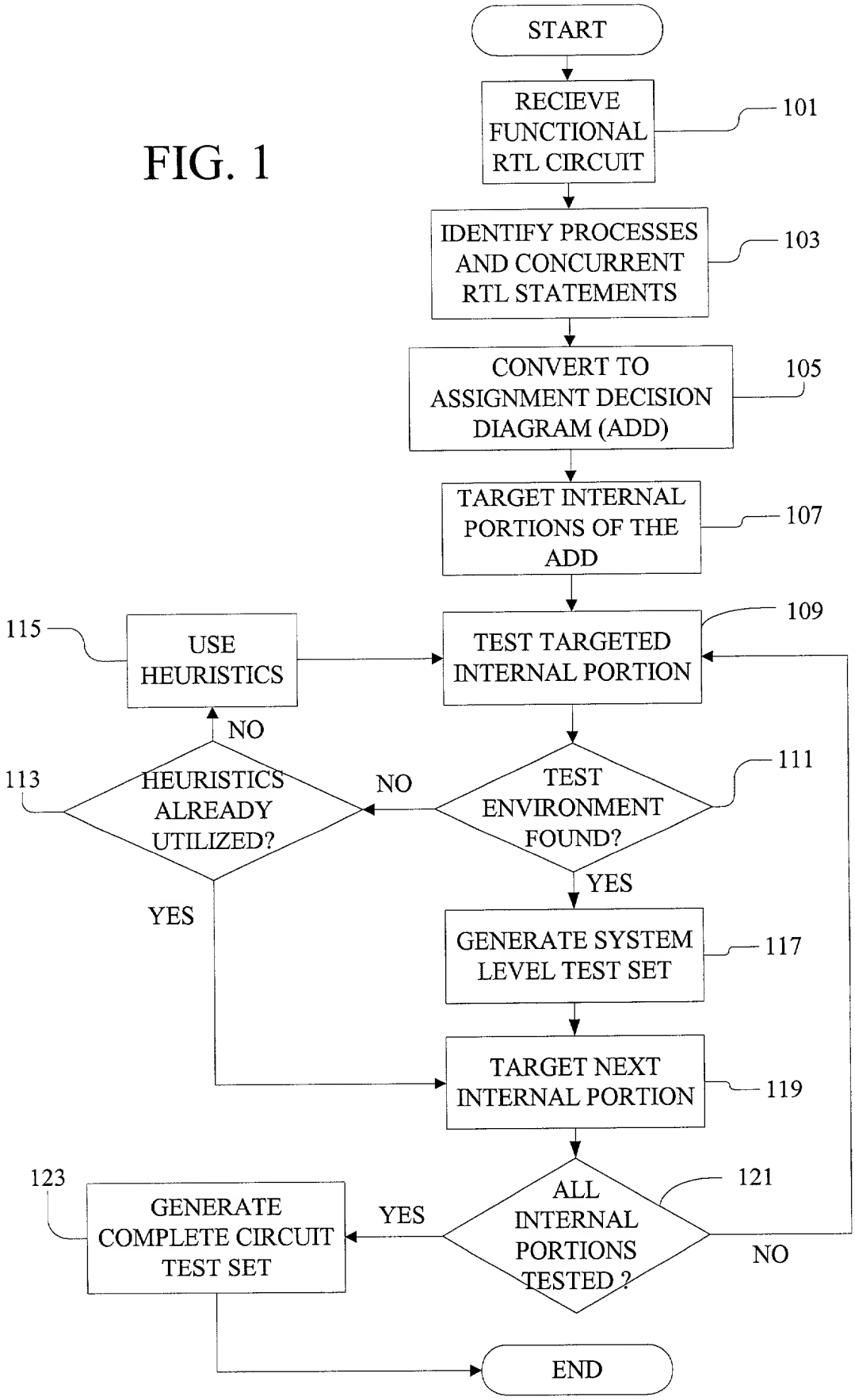
FIG. 1


START
RECIEVE FUNCTIONAL RTL CIRCUIT
101
IDENTIFY PROCESSES AND CONCURRENT RTL STATEMENTS
103
CONVERT TO ASSIGNMENT DECISION DIAGRAM (ADD)
105
TARGET INTERNAL PORTIONS OF THE ADD
107
TEST TARGETED INTERNAL PORTION
109
TEST ENVIRONMENT FOUND?
111
NO
HEURISTICS ALREADY UTILIZED?
113
NO
USE HEURISTICS
115
YES
YES
GENERATE SYSTEM LEVEL TEST SET
117
TARGET NEXT INTERNAL PORTION
119
ALL INTERNAL PORTIONS TESTED ?
121
NO
YES
GENERATE COMPLETE CIRCUIT TEST SET
123
END

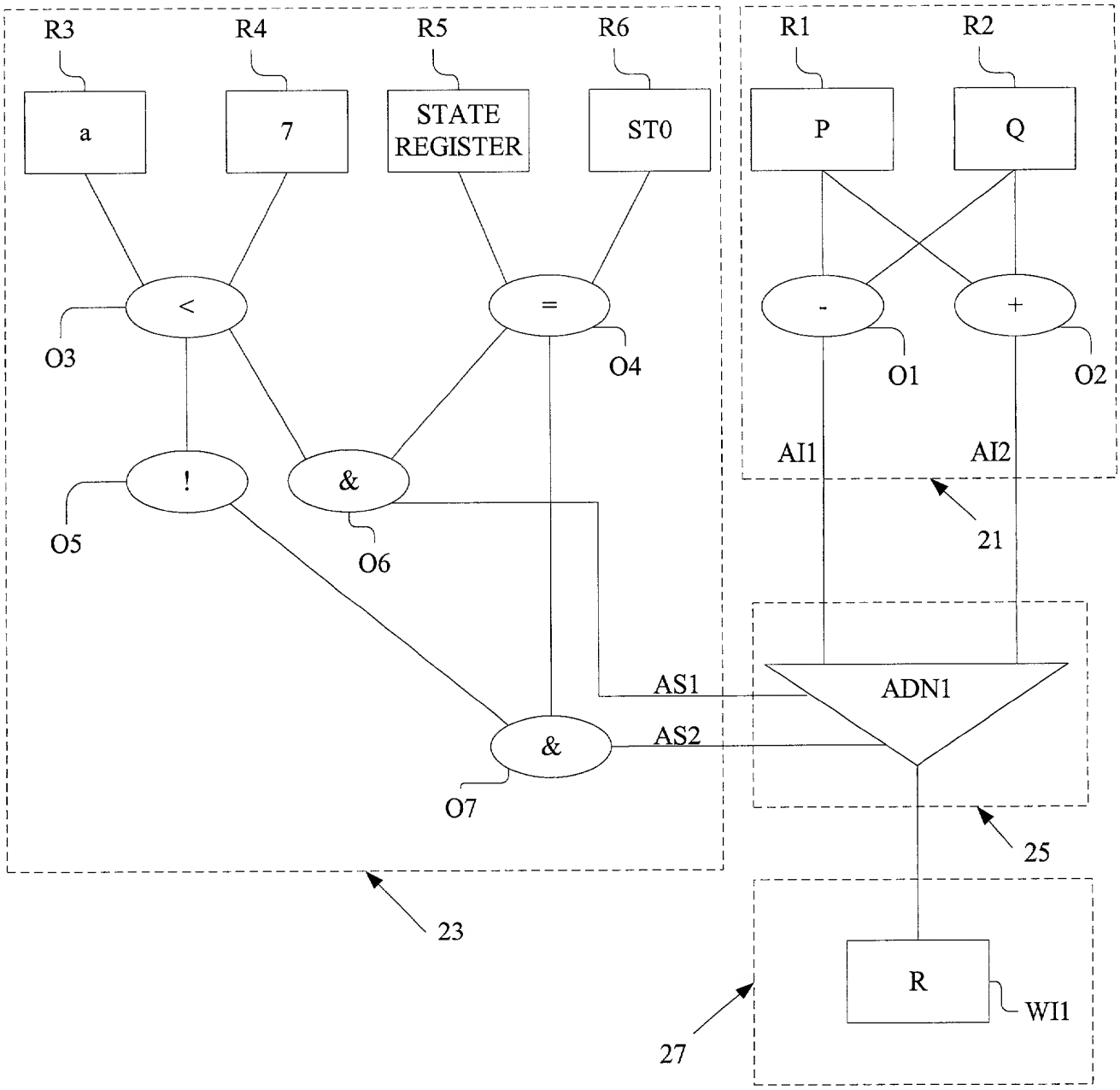
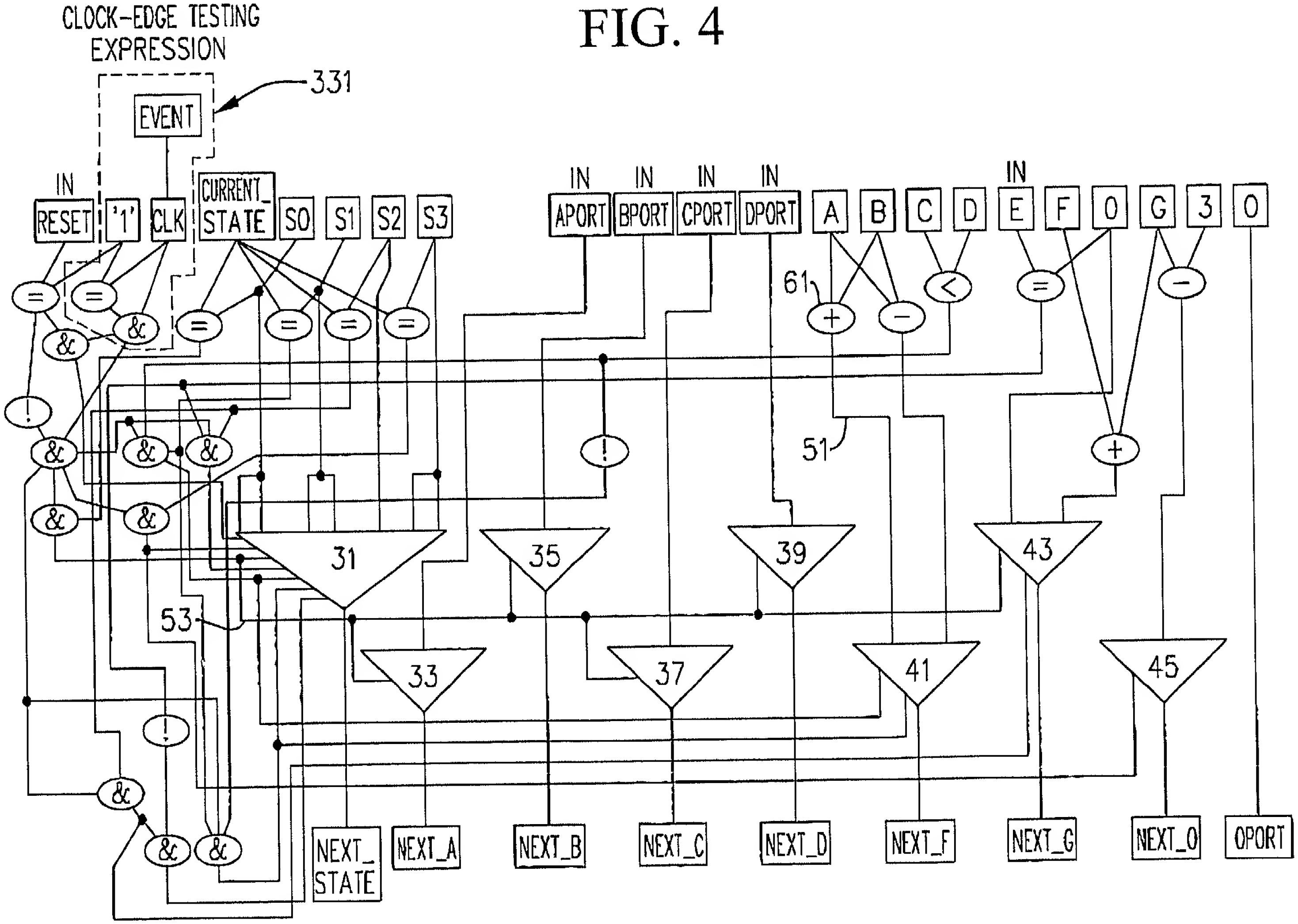

R3
R4
R5
R6
a
7
STATE REGISTER
ST0
<
=
O3
O4
!
&
O5
O6
&
O7
AS1
AS2
23
R1
R2
P
Q
-
+
O1
O2
AI1
AI2
21
ADN1
25
R
WI1
27

FIG. 2

```
entity T1 is
        port (    RESET, CLK :IN std_logic;
                  APORT, BPORT, CPORT, DPORT : IN std_logic_vector (7 downto 0);
                  E : IN std_logic;
                  OPORT : OUT std_logic_vector (7 downto 0) );
        end T1;
architecture RTL of T1 is
        type STATE_TYPE is ( S0, S1, S2, S3);
        signal CURRENT_STATE, NEXT_STATE : STATE_TYPE;
        SIGNAL A, B, C, D, F, G, O, NEXT_A, NEXT_B, NEXT_C, NEXT_D,
                  NEXT_F, NEXT_G, NEXT_0 : std_logic_vector (7 downto 0);
        begin
        COMBIN : process (CURRENT_STATE)
                begin
                        NEXT_A <= A; NEXT_B <= B; NEXT_C <= C; NEXT_D <= D;
                        NEXT_F <= F; NEXT_G <= G; NEXT_O <= O; OPORT <= O;
                        case CURRENT_STATE is
                                when S0 =>
                                        NEXT_A <= APORT; NEXT_B <= BPORT;
                                        NEXT_C <= CPORT; NEXT_D <= D;
                                        NEXT_G <= "00000000";
                                when S1 =>                                          301
                                        if ( C < D ) then
                                                NEXT_F <= A + B;
                                                NEXT_STATE <= S2;
                                        else
                                                NEXT_F <= A - B;
                                                NEXT_STATE <= S3;
                                        end if;
                                when S2 =>
                                        NEXT_G <= F+ G;
                                        if ( E= '0' ) then
                                                NEXT_STATE <= S1;
                                        else
                                                NEXT_STATE <= S3;
                                        end if;
                                when S3 =>
                                        NEXT_O <= G - 3;
                                        NEXT_STATE <= S0;
                        end case;
                end process;
        SYNCH : process (CLK, RESET)
                begin
                        if ( CLK'event and CLK = '1' ) then
                                if ( RESET = '1' ) then
                                        CURRENT_STATE <= S0;
                                else
                                        A <= NEXT_A; B <= NEXT_B; C <= NEXT_C; D <= NEXT_D;
                                        G <= NEXT_G; F <= NEXT_F; O <= NEXT_O;
                                end if;
                        end if;
                end process;
end RTL;
```

FIG. 3

FIG. 4


CLOCK-EDGE TESTING
EXPRESSION
331
EVENT
IN
RESET
'1'
CLK
CURRENT_
STATE
S0
S1
S2
S3
IN
APORT
IN
BPORT
IN
CPORT
IN
DPORT
A
B
C
D
IN
E
F
0
G
3
0
61
51
53
31
33
35
37
39
41
43
45
NEXT_
STATE
NEXT_A
NEXT_B
NEXT_C
NEXT_D
NEXT_F
NEXT_G
NEXT_O
OPORT

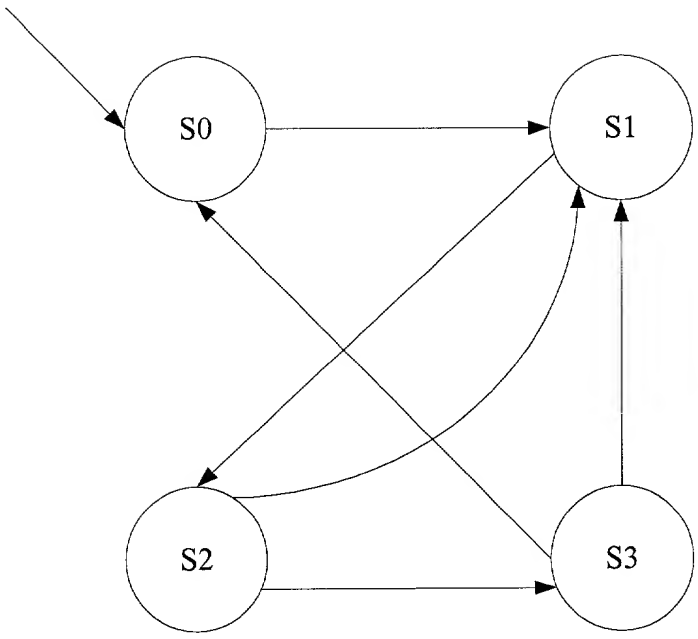


FIG. 5

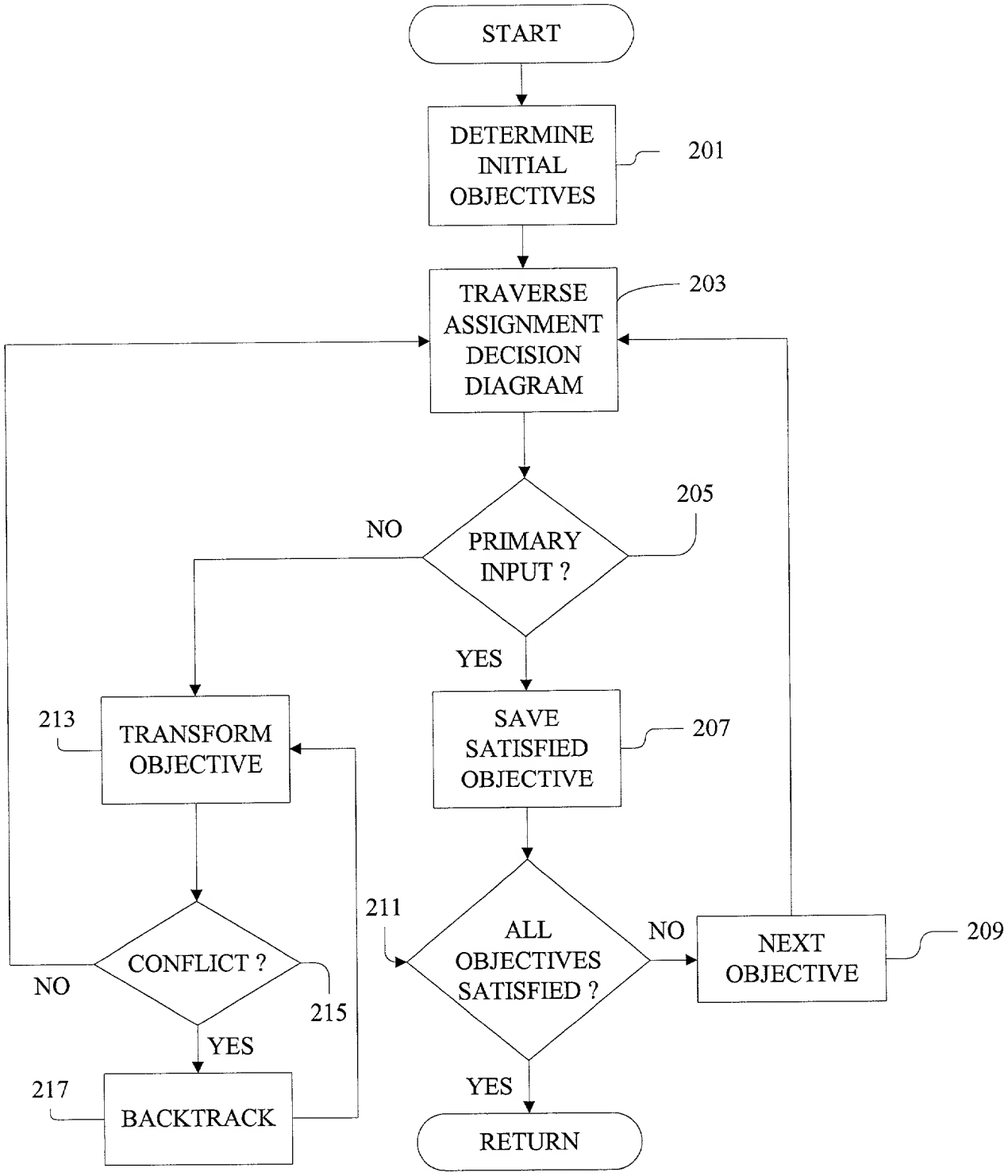

START
DETERMINE INITIAL OBJECTIVES
201
TRAVERSE ASSIGNMENT DECISION DIAGRAM
203
PRIMARY INPUT ?
205
NO
YES
TRANSFORM OBJECTIVE
213
SAVE SATISFIED OBJECTIVE
207
CONFLICT ?
NO
215
YES
BACKTRACK
217
ALL OBJECTIVES SATISFIED ?
211
NO
NEXT OBJECTIVE
209
YES
RETURN

FIG. 6

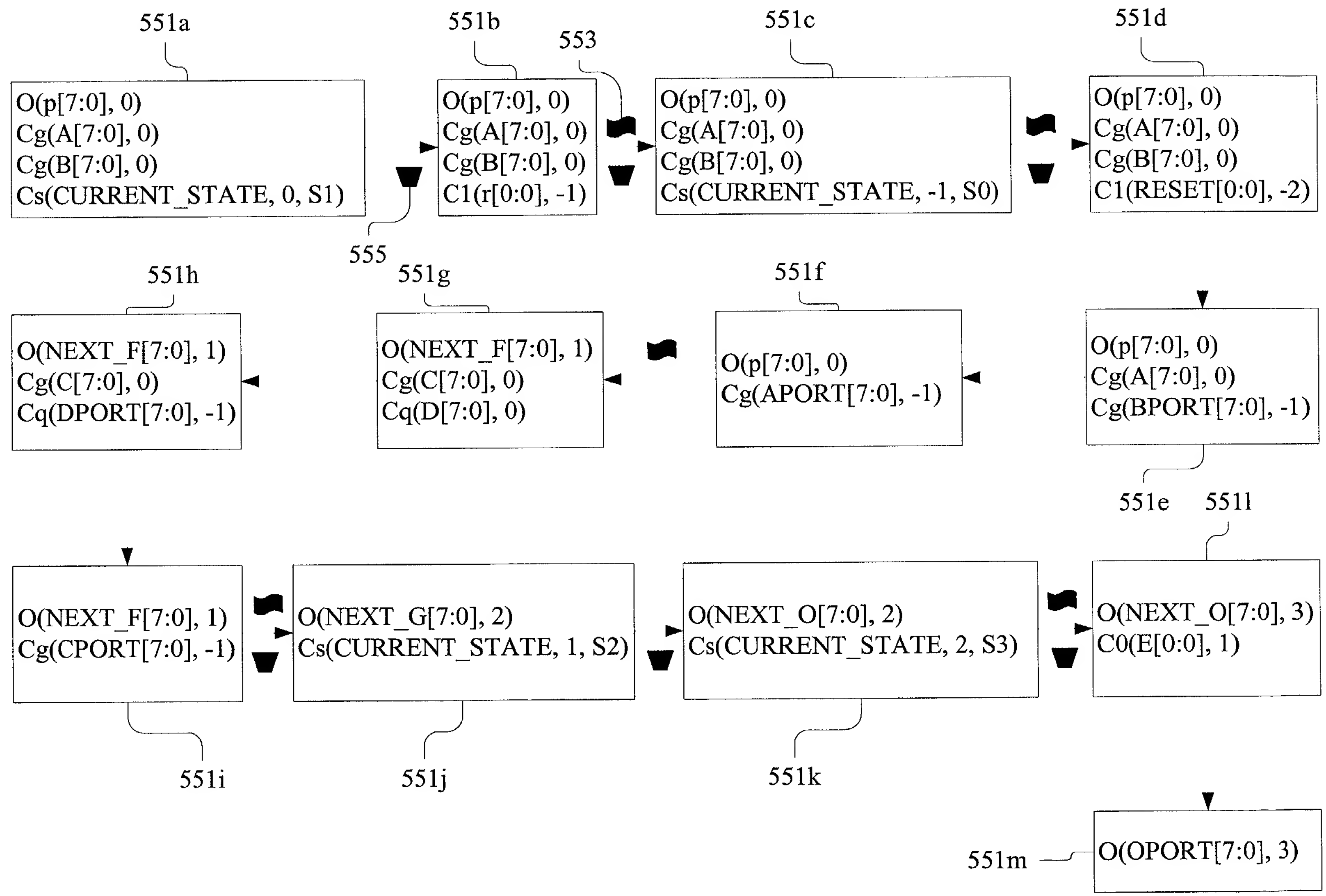

551a
O(p[7:0], 0)
Cg(A[7:0], 0)
Cg(B[7:0], 0)
Cs(CURRENT_STATE, 0, S1)
551b
O(p[7:0], 0)
Cg(A[7:0], 0)
Cg(B[7:0], 0)
C1(r[0:0], -1)
553
555
551c
O(p[7:0], 0)
Cg(A[7:0], 0)
Cg(B[7:0], 0)
Cs(CURRENT_STATE, -1, S0)
551d
O(p[7:0], 0)
Cg(A[7:0], 0)
Cg(B[7:0], 0)
C1(RESET[0:0], -2)
551h
O(NEXT_F[7:0], 1)
Cg(C[7:0], 0)
Cq(DPORT[7:0], -1)
551g
O(NEXT_F[7:0], 1)
Cg(C[7:0], 0)
Cq(D[7:0], 0)
551f
O(p[7:0], 0)
Cg(APORT[7:0], -1)
551e
O(p[7:0], 0)
Cg(A[7:0], 0)
Cg(BPORT[7:0], -1)
551i
O(NEXT_F[7:0], 1)
Cg(CPORT[7:0], -1)
551j
O(NEXT_G[7:0], 2)
Cs(CURRENT_STATE, 1, S2)
551k
O(NEXT_O[7:0], 2)
Cs(CURRENT_STATE, 2, S3)
551l
O(NEXT_O[7:0], 3)
C0(E[0:0], 1)
551m
O(OPORT[7:0], 3)

FIG. 7

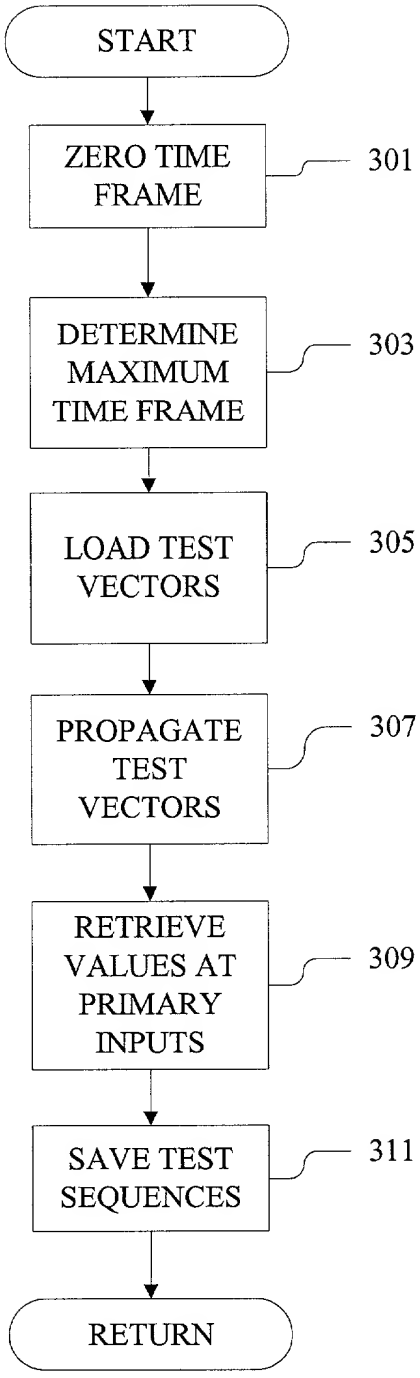

START
ZERO TIME FRAME
301
DETERMINE MAXIMUM TIME FRAME
303
LOAD TEST VECTORS
305
PROPAGATE TEST VECTORS
307
RETRIEVE VALUES AT PRIMARY INPUTS
309
SAVE TEST SEQUENCES
311
RETURN

FIG. 8

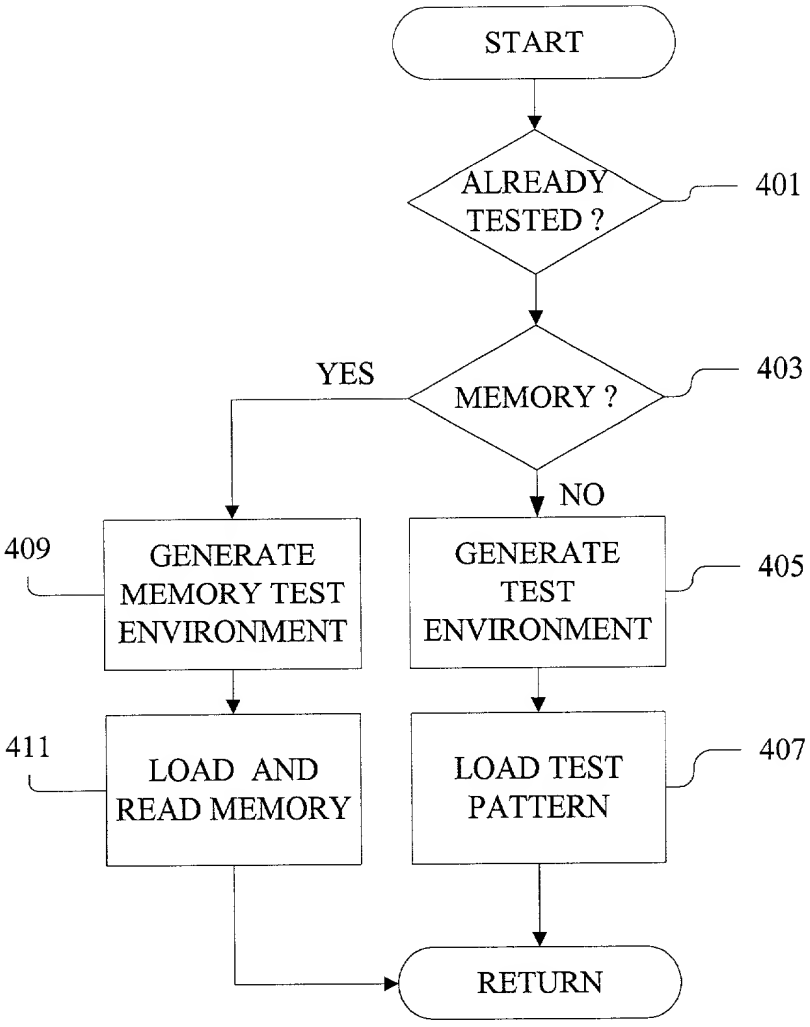

START
ALREADY TESTED ?
401
YES
MEMORY ?
403
NO
409
GENERATE MEMORY TEST ENVIRONMENT
GENERATE TEST ENVIRONMENT
405
411
LOAD AND READ MEMORY
LOAD TEST PATTERN
407
RETURN

FIG. 9

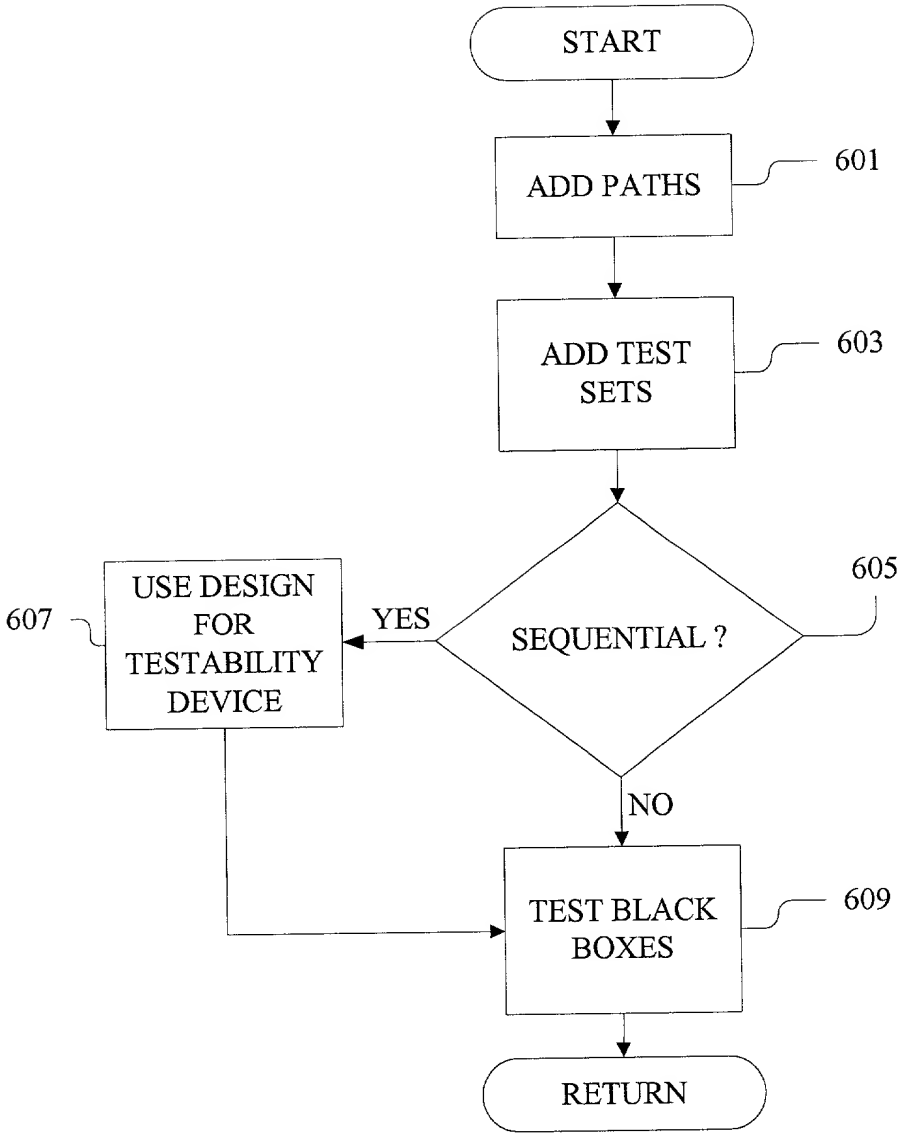

START
ADD PATHS
601
ADD TEST SETS
603
SEQUENTIAL ?
605
YES
USE DESIGN FOR TESTABILITY DEVICE
607
NO
TEST BLACK BOXES
609
RETURN

FIG. 10